UNIVERSITÀ
DEGLI STUDI
DELL'AQUILA

Università degli studi dell'Aquila

Web Engineering Project
Report

AuleWeb

Student :
Dario Di Mambro

Professors :
Giuseppe Della Penna

February 16, 2025

Contents

1 Introduction

This document provides an overview of the development and implementation of the AuleWeb project, created as part of the Web Engineering course for the academic year 2023/2024. The project is developed following the specification on project link , which outlines the requirements for a classroom management system inspired by the existing university platform Aule Univaq.
The AuleWeb system aims to facilitate classroom organization by allowing users to view, manage, and assign events to classrooms within configurable groups. The system includes functionalities such as event scheduling, classroom availability visualization, recurring event handling, and export in CSV. Additionally, it supports administrative management features, allowing authorized users to configure classrooms and manage equipment lists.
The project employs Java Servlets for server-side logic, JavaScript for client-side interactions, and Freemarker as a template engine, ensuring separation between presentation and logic. The system follows web standards to maintain cross-browser compatibility while offering an intuitive user experience. This document outlines the project's key functionalities, software dependencies, database schema, site structure, and development decisions, providing a guide to its design and implementation.

2 Software Dependencies

This project utilizes a combination of server-side and client-side technologies to implement the required functionalities. Below is a structured overview of the dependencies used, including their versions and purposes.

2.1 Server-Side Dependencies

The back-end is developed using Java Servlets, with additional libraries for template rendering, database access, and CSV handling. The dependencies are managed through Maven, and the following libraries are included:

- Freemarker 2.3.33 – A Java-based template engine used to separate logic from presentation, simplifying dynamic page rendering.
- Freemarker Java 8 Extension 3.0.0 – Provides support for Java 8's `java.time` API in Freemarker templates.
- MySQL Connector 8.0.33 – A JDBC driver used to interact with the MySQL database.
- Java Servlet API 4.0.0 – Provides the core functionality for handling HTTP requests and responses in a Java web application.
- JSP API 2.3.0 – Enables the use of JavaServer Pages (JSP) for dynamic web content.
- Expression Language (EL) API 3.0.0 – Used for handling dynamic expressions within JSP pages.
- Java WebSocket API 1.1 – Enables real-time, bidirectional communication between the client and server.

- Java Security API 1.0 – Provides security mechanisms for enterprise applications.
- Java Annotation API 1.3.2 – Required for annotation-based configurations such as `@Resource` and `InitialContext`.
- Apache Commons CSV 1.12.0 – A library for handling CSV file operations, used for exporting event data.

The project is built and managed using Maven, with the Maven Compiler Plugin 3.13.0 ensuring compatibility with Java 14.

2.2 Client-Side Dependencies

The front-end of the application is developed using HTML5, CSS, JavaScript, and Freemarker. Additionally, the following libraries and frameworks are utilized:

- jQuery – A lightweight JavaScript library used for DOM manipulation, AJAX requests, and event handling.
- Freemarker – Used for rendering dynamic HTML templates on the server side before sending them to the client.

The front-end design follows modern web development practices, ensuring accessibility and responsiveness where applicable.

2.3 Build and Deployment Tools

The project is packaged as a WAR (Web Application Archive) and deployed on Apache Tomcat, which provides the necessary runtime for Java Servlets and JSP pages. The Maven WAR Plugin 3.4.0 is used to facilitate WAR packaging.

3 Functionalities Implemented (and Not Implemented)

3.1 Functionalities Implemented

The AuleWeb system was developed following the provided specification, implementing all core functionalities required. The following features have been successfully implemented:

- Display classrooms by group.
- Show events associated with a specific classroom in a given week.
- Show events associated with each classroom on a given day.
- Show all current events (currently in progress) and those scheduled for the next three hours.
- Show events associated with a specific course in a given week.
- Export all events in a given time interval, optionally filtered by course, in CSV format.

- Manage administrators who can:
  - Insert and modify classroom configurations and related groups.
  - Insert and edit events, including recurring events.
  - Manage the list of available equipment.
  - Export and import classroom configurations in CSV format.

All implemented functionalities ensure smooth interaction between the user interface and back-end services, allowing effective classroom and event management.

3.2 Not Implemented or Partially Implemented

The only feature explicitly marked as optional in the project specification was the ability to export events in iCalendar format. Since the specification allows choosing between CSV and iCalendar, this project opted to support CSV export, leveraging the `Apache Commons CSV` library for structured data output.

No additional functionalities were omitted or left incomplete.

The activity diagram 1 visually represents the structure and relationships between different pages of a website.

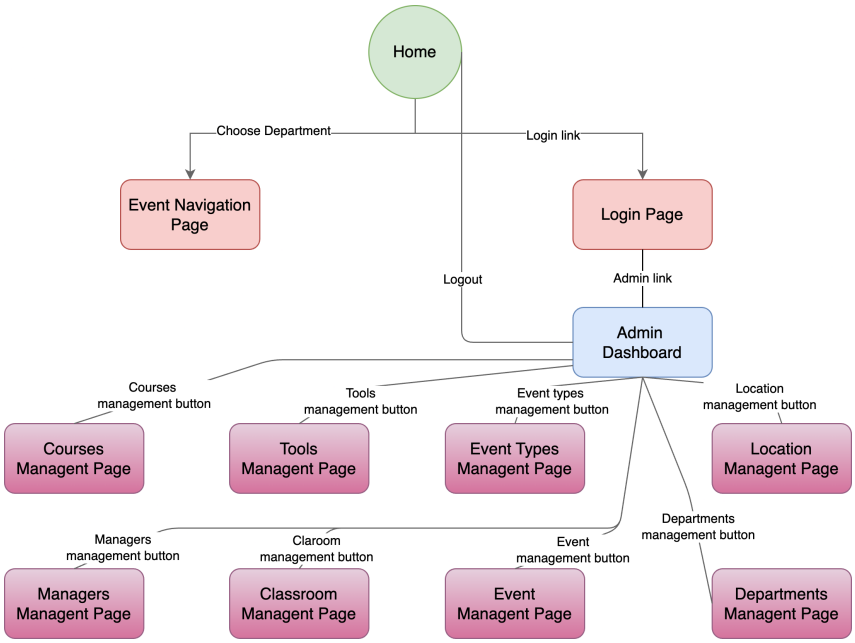

Figure 1: Navigation Diagram

4 Relational Schema of the Database

The database for the AuleWeb system is structured using MySQL and follows a relational model designed to manage classrooms, events, and administrative configurations efficiently. Below is a summary of the key tables and their roles in the system.

4.1 Core Entities

- `classroom` – Represents classrooms, including attributes such as name, capacity, location, manager email, available equipment, and related group.
- `classroom_group` – Defines logical groupings of classrooms (e.g., by department), with a name and description.
- `location` – Stores details about physical locations, including place, building, and floor.
- `manager` – Contains event managers' names and emails, responsible for scheduling events.
- `course` – Lists university courses, referenced by events of type lecture or exam.
- `type` – Defines event categories, such as lecture, final exam, seminar, or meeting.
- `users` – Manages system users, storing authentication credentials.

4.2 Event Management

- `event` – Records events assigned to classrooms, storing details such as date, start and end time, description, associated course (if applicable), event type, and manager.
- `classroom_tool` – Maps available tools to classrooms, enabling the association of equipment like projectors, microphones, and audio systems.

4.3 Relationships and Constraints

- `classroom` references `classroom_group` and `location`, establishing logical and physical associations.
- `event` maintains foreign key constraints to `classroom`, `manager`, `course`, and `type`, ensuring integrity in scheduling.
- `classroom_tool` enforces a many-to-many relationship between `classroom` and `tool`, ensuring accurate equipment tracking.
- Foreign keys with `ON DELETE SET NULL` or `ON DELETE CASCADE` ensure that related records remain consistent upon deletions.

The relational diagram figure 2 illustrates the connections between these tables.

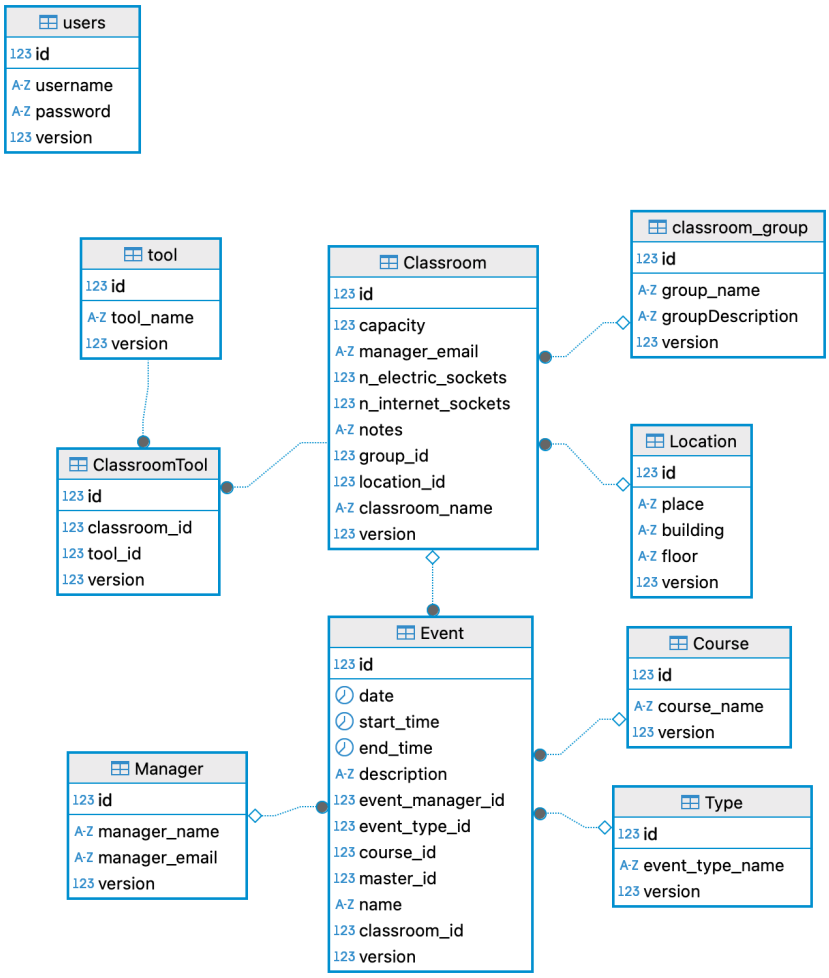

Figure 2: Relation Schema Diagram

5 Analytical Description of the Site Layout

The layout of the AuleWeb system is structured to provide an intuitive and efficient user experience, ensuring easy navigation and interaction with the classroom scheduling functionalities. The system includes both a general user interface for browsing events and an administrative interface for managing the database.

5.1 Overall Structure

The website follows a structured layout consisting of the following key sections:

- **Header** – Contains:
  - A link to the homepage.
  - A dynamic page title that updates based on the currently open page.
  - An icon that provides access to authentication options:
    - If the user is not logged in, it links to the login page.
    - If the user is logged in, it displays links to the admin page and a logout option.

- **Main Content** – Varies depending on the selected page, with dynamic sections for:
  - Selecting and viewing departmental calendars.
  - Viewing classroom events week by week, moving forward or backward.
  - Viewing all current and recently started events.
  - Viewing events scheduled for a specific day.
  - Viewing events for a specific course, navigating week by week.
  - Exporting a CSV file containing events within a specified date range.
- **Navigation and Interaction**:
  - The homepage dynamically presents department selection options.
  - When navigating between weeks in a classroom or course view, the URL is updated via a `GET` request, and the database is queried again to update the displayed events.
  - For specific day events and currently active events, an AJAX `fetch` request is used to update content dynamically without requiring a full page reload.

5.2 Administrative Interface

Administrators have additional functionality available after logging in. The admin workflow consists of the following:

- **Login Page** – Admin users authenticate through a dedicated login form.
- **Admin Dashboard** – After logging in, administrators gain access to a control panel with buttons leading to different database management pages.
- **Database Interaction via Forms** – The admin pages provide user-friendly interaction with the database through dynamically populated forms:
  - Forms initially display only selection options (e.g., a list of classrooms).
  - When an admin selects an entity (e.g., a classroom), relevant fields (such as location, manager, capacity, etc.) are dynamically populated with the current data from the database, making it easier to edit or delete existing records.
  - If the admin is adding a new entity, the fields remain empty for manual input.
  - For event deletion or modification, an admin can specify a date range to filter events dynamically, making event selection easier.
- **Classroom Data Export** – From the main admin page, administrators can export classroom data in CSV format.

5.3 Static and Dynamic Components

- **Static Elements**:
  - The header and authentication controls remain consistent across all pages.
  - The homepage structure is static, while department options are populated dynamically.
- **Dynamic Elements**:
  - Event listings are dynamically retrieved from the database and updated based on user navigation.
  - The page content updates based on user interactions, such as selecting a department, changing the week, or filtering events.
  - The admin interface retrieves and updates data dynamically, ensuring efficient database interaction.
  - Forms in the admin interface populate dynamically when an entity is selected, reducing manual input and improving usability.

5.4 Styling and Layout Approach

The AuleWeb system uses a structured layout to ensure clarity and usability:

- CSS – Styles the user interface, ensuring readability and responsiveness.
- Freemarker Templates – Used for rendering dynamic HTML content, ensuring separation between logic and presentation.
- JavaScript and jQuery – Enhance interactivity, particularly for dynamic content updates via `fetch` requests.

6 Cross-Browser Programming & Rendering

6.1 List of Compatible Browsers

The AuleWeb system was tested on the following browsers to ensure proper functionality and consistent rendering:

- Edge (latest version at the time of testing)
- Chromium-based browsers, including:
  - Chrome (latest version)
  - Brave (latest version)
- Safari (latest version on macOS)

6.2 Encountered Issues

During testing, no significant rendering or functionality issues were found across the tested browsers. The application maintains a consistent appearance and behavior on Edge, Chromium-based browsers, and Safari.

6.3 Degradation

The AuleWeb system relies on JavaScript for key functionalities, including:

- Dynamic event retrieval via AJAX requests (`fetch` API)
- Form population for admin features
- Page updates based on user interaction (e.g., week navigation)

If JavaScript is disabled, core functionalities such as dynamic event loading, AJAX-based updates, and interactive form population will not work. While basic navigation and static content remain accessible, the user experience will be significantly limited.

7 Screenshots

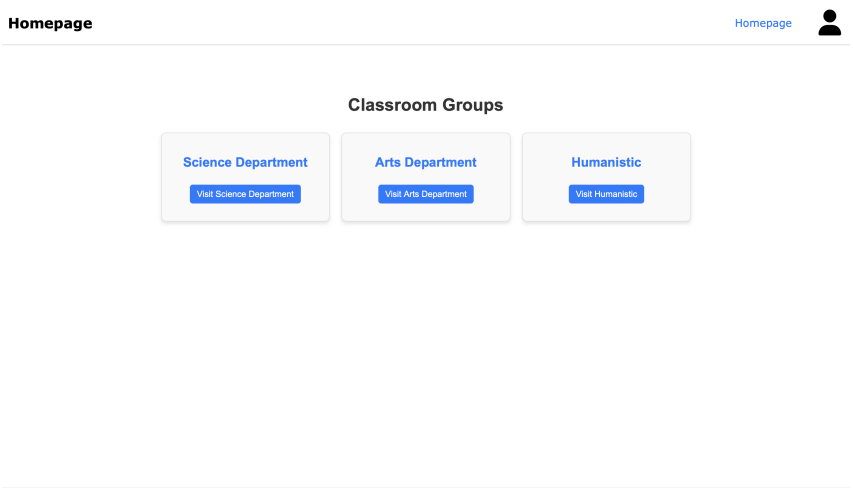

Figure 3: Homepage

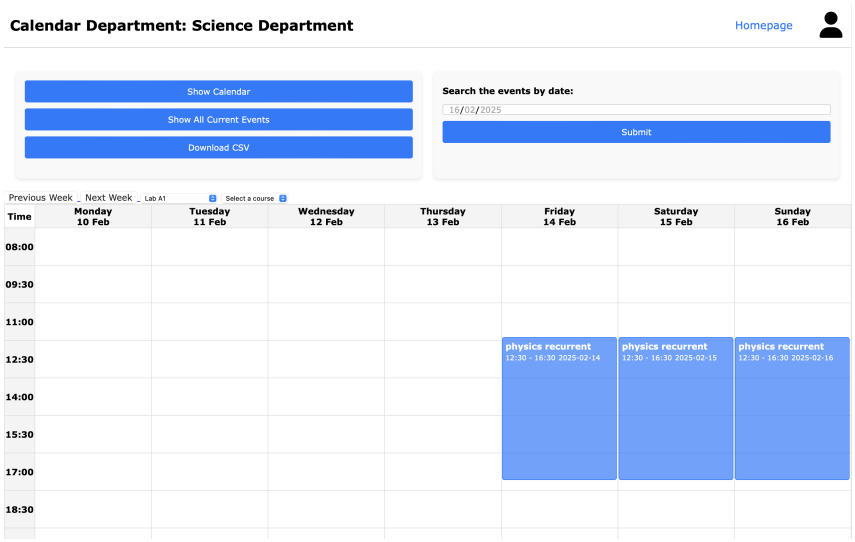

Figure 4: Calendar

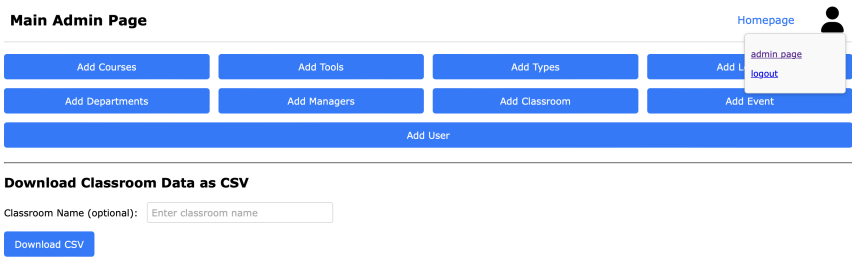

Figure 5: Admin Page

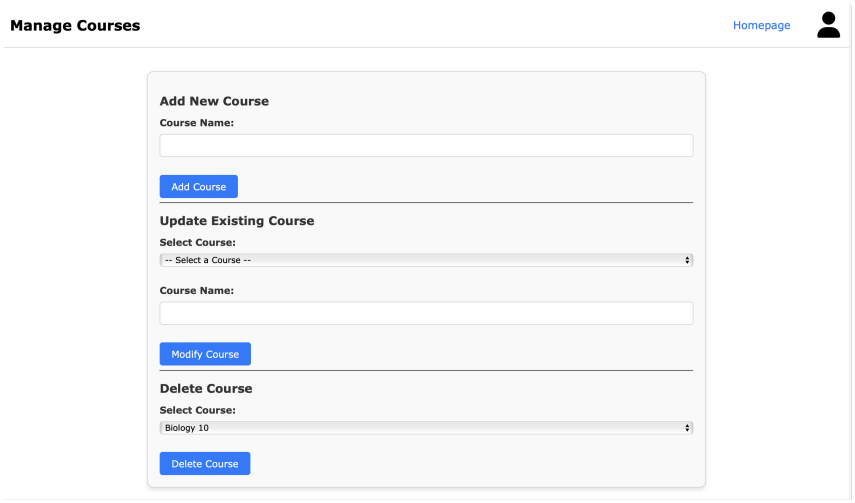

Figure 6: Course Management Page